Amendments to the Drawings:

The attached sheet of annotated drawings includes changes to Fig. 1A. The replacement sheet, which includes Fig. 1A, replaces the original sheet including Fig. 1A.

Attachment: Replacement Sheet
Annotated Sheet Showing Changes

REMARKS/ARGUMENTS

Claims 1-31 and 43-48 are pending, claims 1, 3, 14, 16, 27, 28, and 45 are amended.

The specification is amended to include some examples of the AEP verification tools from www.parasoft.com. Since the entire contents of the website, as of the filing date of this application, is expressly incorporated by reference on page 8, lines 7-9, no new matter is added.

Additionally, the paragraph beginning on page 5, line 10 is amended to describe what is shown (and described) in FIG. 1A. No new matter is added.

Also, FIG. 1A is amended to include reference numeral numbers for clarifications.

The specification is objected to because it "is devoid of terms such as 'verification programs,' rather, it "discloses 'verification tools.'" Applicant respectfully submit that one skilled in the art of computer software would readily understand that a "verification tool" is the same as a "verification programs." In deed, Wikipedia clearly states that a "programming tool or software development tool is a program or application that software developers use to create, debug, maintain, or otherwise support other programs and applications. (Wikipedia.org, underlining added.). Furthermore, the added description of some examples of the AEP verification tools from www.parasoft.com makes it very clear that the verification tools are independent computer programs. In view of the above remarks, it is respectfully requested that the above objection be withdrawn.

Claims 1-26 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. In particular, the Examiner states that there is no support in the specification for "executing a first software verification program corresponding to a first lifecycle phase of the computer software, wherein the first software verification program automatically generates one or more test cases from the source code of the computer software," and "executing a second software verification program corresponding to a second lifecycle phase of the computer software different from the first lifecycle phase," in claims 1 and 14. Applicant respectfully disagrees.

The specification is very clear about "The AEP tools are capable of generating their own test cases, without any prior knowledge of the code" (page 8, lines 4-5), "The verification tools in the AEP process are automated. That is the tools automatically generate test cases" (page 12, lines 20-21), and "these verification tool [sic] are capable of automatically generating one or more test cases" (cited, page 14, lines 13-14). Furthermore, the specification describes:

> AEP implementation can begin at **any stage of an organization's software lifecycle** and extend through all subsequent phases and cycles. The organization begins implementing the AEP practices **relevant to their current lifecycle phase** (as shown in FIG. 1A), and then uses these practices to gauge whether the team and software should proceed to the next lifecycle phase. **When the next lifecycle phase begins**, the organization implements the **AEP practices related to that lifecycle phase**, and then repeats the process again.
>
> (Page 6, line 27 to page 7, line 1, bolding is added for emphasis).

Additionally, "AEP practices" as used through out the specification and shown in FIG. 1A includes executing the verification tools. For example, "AEP practices and technologies expose different types of mistakes and errors at the earliest possible phase of the software lifecycle" (page 5, lines 14-15); "AEP makes this possible by providing an infrastructure that automatically executes the appropriate practices at the appropriate times . . . These practices are performed in the background" (Page 6, lines 13-15); "The organization begins implementing the AEP practices relevant to their current lifecycle phase (as shown in FIG. 1A), and then uses these practices to gauge whether the team and software should proceed to the next phase" (Page 6, lines 28-31); and "Problems discovered at this stage usually indicate the need for additional AEP practices somewhere in the software development lifecycle" (page 8, lines 22-23).

Therefore, it is respectfully submitted that the specification supports the claimed language of "executing a first software verification program corresponding to a first lifecycle phase [a current lifecycle] of the computer software, wherein the first software verification program automatically generates one or more test cases from the source code of the computer

software," and "executing a second software verification program corresponding to a second lifecycle phase ["when a next life cycle begins"] of the computer software different from the first lifecycle phase."

In view of the above remarks, it is respectfully requested that the above rejections be withdrawn and the above limitation be considered for the purpose of the examination.

Claims 1-5, 7-18, 20-30 and 43-48 are rejected under 35 U.S.C. 102(e) as being anticipated by Jorapur (U.S. 7,299,382). Claims 6, 19, and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Jorapur in view of Man et al. (U.S. 6,625,760). Applicant submits that all of the claims currently pending in this application are patentably distinguishable over the cited references for the following reasons, and reconsideration and allowance of this application are respectfully requested.

Amended independent **claims 1 and 14** include, among other elements, "executing a plurality of software verification programs to verify the computer software, wherein each of the plurality of software verification tools relates to a respective lifecycle phase of the computer software and automatically generates one or more test cases from the source code of the computer software, wherein each of the plurality of software verification programs is an independent verification tool including a user interface and is capable of being executed in a batch process," "executing a first software verification program corresponding to a first lifecycle phase of the computer software, wherein the first software verification program automatically generates one or more test cases from the source code of the computer software," and "executing a second software verification program corresponding to a next lifecycle phase of the computer software different from the first lifecycle phase." Jorapur does not teach the above elements.

First, Jorapur does not teach the limitation of "wherein each of the plurality of software verification programs is an independent verification tool including a user interface and is capable of being executed in a batch process." Rather, the cited tests 302 of Jorapur are generated by a single test generator 301. "The test generator may be a script, an application, or other element configured to generate tests for software. The test generator 301 may return one or more tests

302 corresponding to the application 300. In one embodiment, the test generator 301 produces tests 302 corresponding to modules that are part of the application 300. For example, the test generator 301 may access one or more modules of the application 300 and generate tests 302 including test code configured to interact with at least one of the modules. (Col. 6, lines 41-52, and FIG. 3, emphasis added.). Therefore, it is clear that the cited tests 302 of Jorapur: a) are NOT "an independent verification tool," b) do NOT include "a user interface," and c) are NOT "capable of being executed in a batch process."

Second, regarding the limitation of "each of the plurality of software verification programs . . . automatically generates one or more test cases from the source code," the examiner cites to col. 4:55-65; col. 6:52-60; and col. 9:45-55 of Jorapur as teaching this limitation. Applicant respectfully disagrees. The above cited passages of Jorapur simply teaches different tests (e.g., test cases) that may be generated in one or more blocks corresponding to one or more parts of the program under test. These tests 302 are generated by a single test generator 301. "For example, the test generator 301 [not each of the tests 302] may access one or more modules of the application 300 and generate tests 302 including test code configured to interact with at least one of the modules. (Col. 6, lines 43-52, and FIG. 3, emphasis added.). This means that, at best, the tests 302 are the generated test cases generated by the single test generator 301.

The Examiner states that "because all is required by this limitation is that the fact that the software verification programs tests the software errors." (Office action, page 6, middle paragraph.). However, Applicant respectfully submit that the limitation requires that "each of the plurality of software verification programs. . . automatically generates one or more test cases from the source code." However, it is clear from Jorapur's disclosure (e.g., see the above cited text) that the individual tests 302 do not and can not "automatically generate one or more test cases." Rather, at best, they may correspond to the claimed "generated test cases," and NOT generating the test case by themselves. Instead, any test cases in Jorapur is generated by the single test generator 302. Consequently, Jorapur does not teach "each of the plurality of software verification programs. . . automatically generates one or more test cases from the source code."

Third, Jorapur does not teach the limitation of "executing a first software verification program corresponding to a first lifecycle phase of the computer software, wherein the first software verification program automatically generates one or more test cases from the source code of the computer software," and "executing a second software verification program corresponding to a next lifecycle phase of the computer software different from the first lifecycle phase wherein the second software verification program automatically generates one or more test cases from the source code of the computer software." The cited language of "errors or failures may be detected and reported as part of the testing framework at any stages (lifecycle phases)" does not teach a) executing a first and second software verification program corresponding to a first lifecycle phase and a different lifecycle phase different from the first life cycle; or a next lifecycle phase. Rather, the tests 302 of Jorapur may be run only at a single lifecycle phase (not different one) or another lifecycle phase, which is not the "next lifecycle phase." Additionally, as described above, Jorapur does not teach "wherein the first software verification program automatically generates one or more test cases," or "wherein the second software verification program automatically generates one or more test cases from the source code of the computer software," because the tests 302 of Jorapur do NOT generate test cases, rather, the single test generator 301 generates the test 302 (the alleged test cases).

Consequently, for at least any one of the above three reasons, Jorapur does not teach all elements of amended claims 1 and 14 and therefore, claims 1 and 14 are not anticipated by Jorapur .

Independent **claims 27 and 45** include, among other elements, "wherein each of the plurality of software verification programs is an independent verification tool including a user interface and is capable of being executed in a batch process," "determining what phase of the lifecycle the detected error was introduced, based on analyzing the detected error," and ""executing the plurality of software verification programs to verify the class of the detected error is detected in a same lifecycle phase of the computer software as the determined lifecycle phase wherein the detected error was introduced." Jorapur does not teach the above elements.

First, regarding the element of "wherein each of the plurality of software verification programs is an independent verification tool including a user interface and is capable of being executed in a batch process," as explained above this limitation is not disclosed by Jorapur.

Second, the element of "determining what phase of the lifecycle the detected error was introduced, based on analyzing the detected error" is also not disclosed by Jorapur. There is no disclosure in Jorapur regarding "determining what phase of the lifecycle the detected error was introduced." The Examiner cites that the language of "errors or failures may be detected and reported as part of the testing framework at any stage" (lifecycle phase). (Office action, page 7, last paragraph). It is true that errors may be detected and reported at any stage (lifecycle phase), in Jorapur. However, given a detected error (which may have been introduced at any lifecycle phase not known at the time of detection), analyzing it and determining what phase of the lifecycle it was introduced is NOT disclosed by the above cited text or any part of Jorapur.

Third, with respect to "executing the plurality of software verification programs to verify the class of the detected error is detected in a respective lifecycle phase of computer software" Jorapur does not teach any "class of known error," rather, Jorapur is directed to a plurality of individual tests 302 for detecting individual specific errors and not a class of errors.

Fourth, Jorapur does not teach "executing the plurality of software verification programs to verify the class of the detected error is detected in a same lifecycle phase of the computer software as the determined lifecycle phase wherein the detected error was introduced." Again, in Jorapur "errors or failures may be detected and reported as part of the testing framework at any stage" (lifecycle phase). There is no teaching of "verify[ing] the class of the detected error is detected in a same lifecycle phase of the computer software as the determined lifecycle phase wherein the detected error was introduced," because Jorapur does not determine (remember) which phase the error was detected.

As a result, for at least each of the above four reasons, Jorapur does not teach all elements of claims 27 and 45 and thus, claims 27 and 45 are not anticipated by Jorapur either.

Dependent **claims 3, 16 and 28** include the additional limitation of "customizing a verification scope of one or more of the verification programs by modifying the common configuration file responsive to an objective criterion of quality of the computer software, wherein the objective criterion of quality is a quality rating of the entire computer software that takes into account [the] verification results of each of the verification programs, the number of test cases executed, and the success or failure of the test cases." Jorapur does not teach this limitation either.

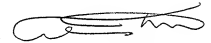
The Examiner points to col. 11, lines 40-50 of Jorapur, more specifically to the language of "different attributes may be specified in a configuration file" and "configurations may be changed" in col. 10, lines 1-15, with respect to previous claims 3, 16 and 28. Applicant respectfully disagrees, There is no teaching in the above cited text about "responsive to an objective criterion of quality of the computer software." More importantly, there is no teaching in the above cited text about "wherein the objective criterion of quality is a quality rating of the entire computer software that takes into account [the] verification results of each of the verification programs, the number of test cases executed, and the success or failure of the test cases."

Accordingly, dependent claims 3, 16 and 28 are also independently patentable over Jorapur as being dependent from an allowable independent claim and for the above additional limitation they include therein.

Dependent claims 2-13, 15-26, 28-31, 43-44, and 46-48 are dependent from allowable independent claims 1, 14, 27, and 45, respectively and therefore include all the limitations of their base claims and additional limitations therein. Accordingly, these claims are also allowable over the cited references, as being dependent from an allowable independent claim and for the additional limitations they include therein.

In view of the foregoing remarks, it is respectfully submitted that this application is now in condition for allowance, and accordingly, reconsideration and allowance are respectfully requested.

Respectfully submitted,
CHRISTIE, PARKER & HALE, LLP

By 
Raymond R. Tabandeh
Reg. No. 43,945
626/795-9900

RRT/clv

CLV PAS839152.1-*-03/3/09 5:12 PM